计算机学院专业必修课

计算机组成

基本认识、数制

高小鹏

北京航空航天大学计算机学院
系统结构研究所

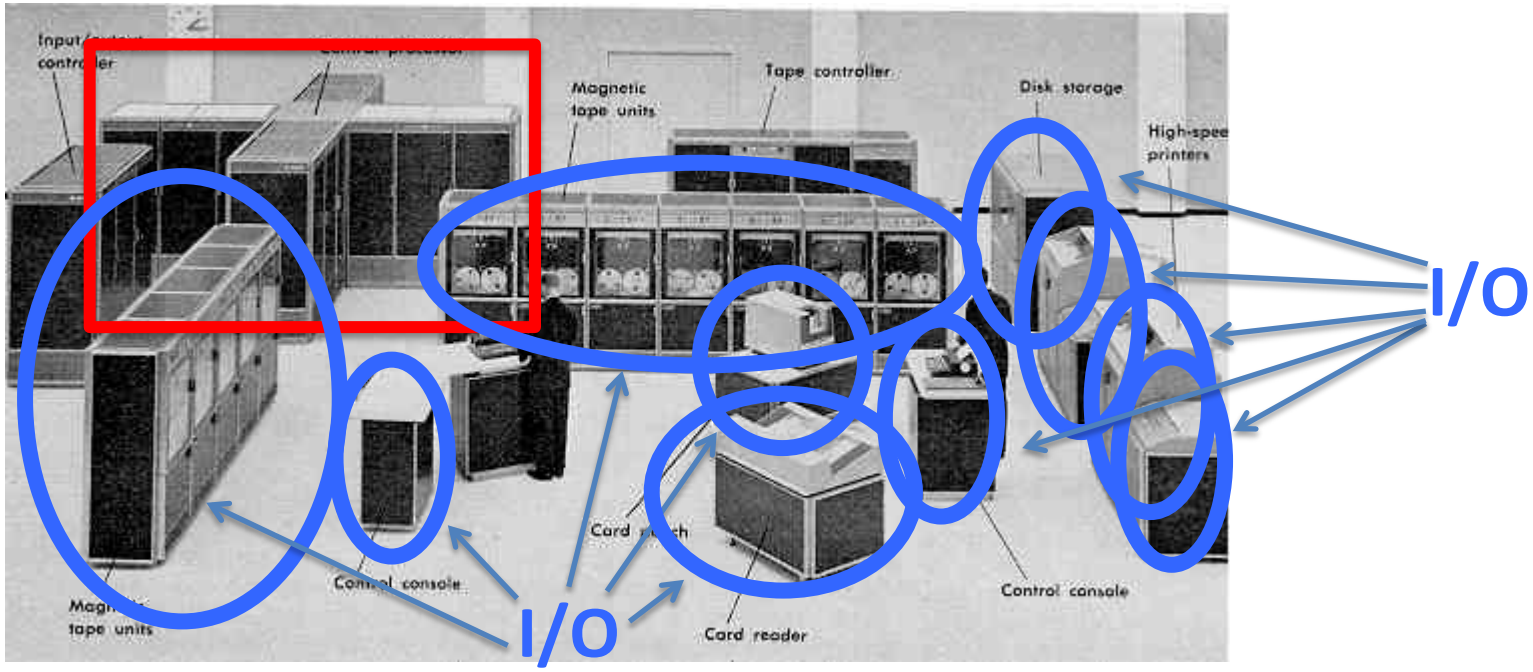
提纲

- 内容主要取材于CS61C第1讲
  - http://inst.eecs.berkeley.edu/~cs61c/su12
- 计算机基本认识
- 数制概念
  - Number Bases(数的进制)
  - Signed(有符合) vs. Unsigned(无符号)
  - Sign Extension(符合扩展)

提纲

- 内容主要取材于CS61C第1讲
  - http://inst.eecs.berkeley.edu/~cs61c/su12
- 计算机基本认识
- 数制概念
  - Number Bases
  - Signed vs. Unsigned
  - Sign Extension

Mainframe Era: 1950s - 1960s



Enabling Tech: Computers

Big Players: "Big Iron" (IBM, UNIVAC)

Cost: $1M, **Target:** Businesses

Using: COBOL, Fortran, timesharing(分时) OS

Minicomputer Era: 1970s

Enabling Tech: Integrated circuits(集成电路)

Big Players: Digital, HP

Cost: $10k, **Target:** Labs & universities

Using: C, UNIX OS

PC Era: Mid 1980s - Mid 2000s

Enabling Tech: Microprocessors

Big Players: Apple, IBM

Cost: $1k, **Target:** Consumers (1/person)

Using: Basic, Java, Windows OS

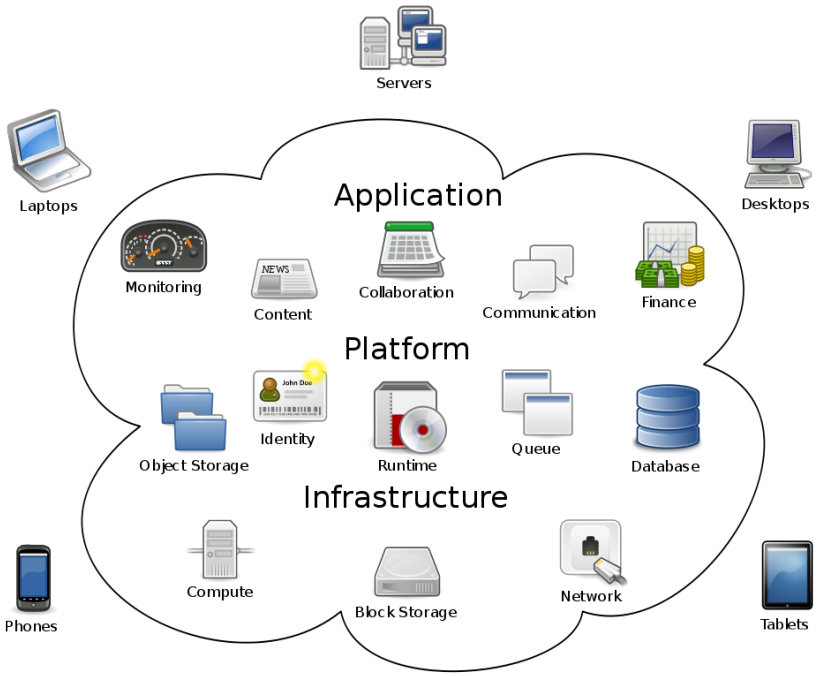
Post-PC Era: Late 2000s - ???

Personal Mobile Devices (PMD):

Enabling Tech: Wireless networking, smartphones
Big Players: Apple, Nokia, …
Cost: $500, **Target:** Consumers on the go
Using: Objective C, Android OS

Post-PC Era: Late 2000s - ???

Cloud Computing:

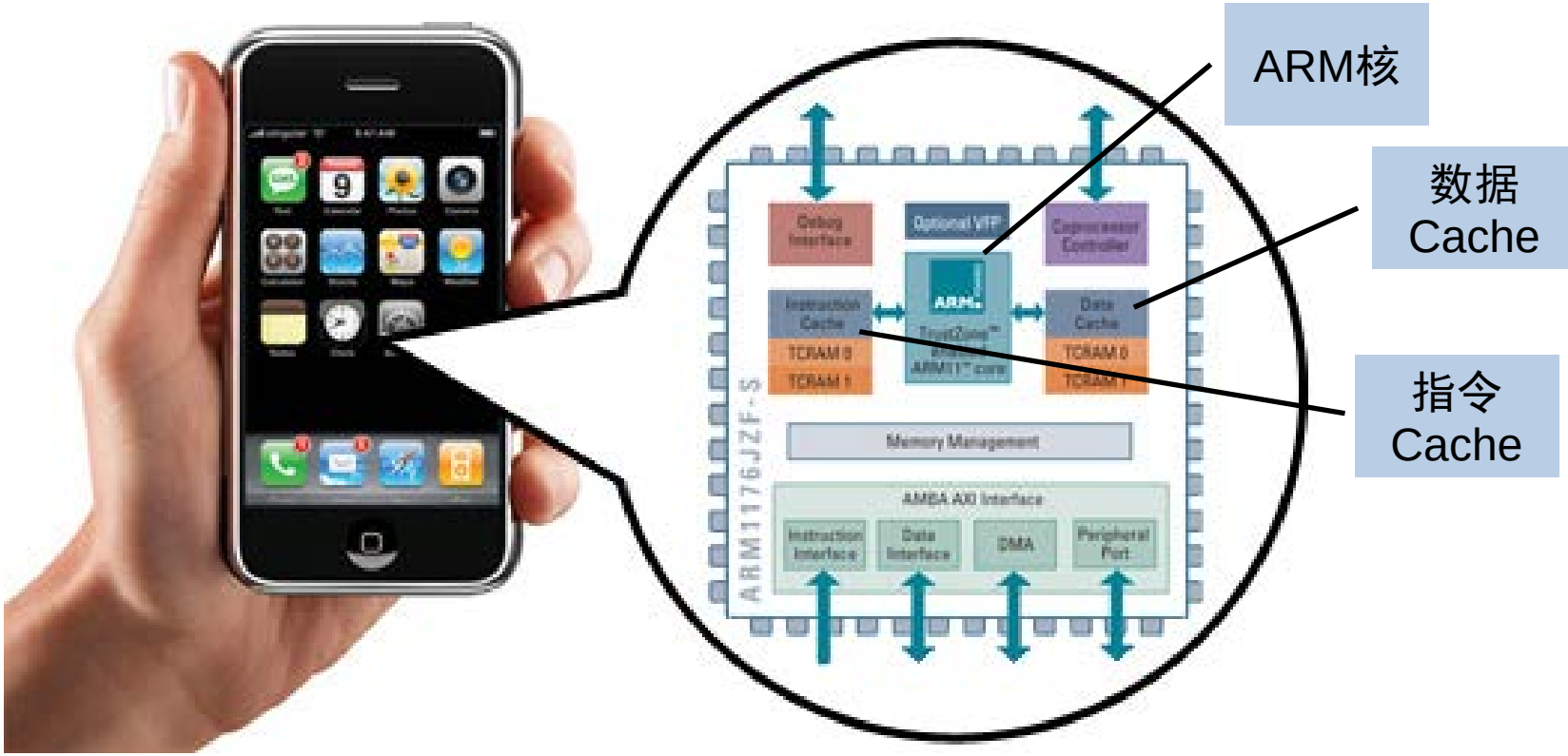


Enabling Tech: Local Area Networks, broadband Internet

Big Players: Amazon, Google, …

Target: Transient users or users who cannot afford high-end equipment

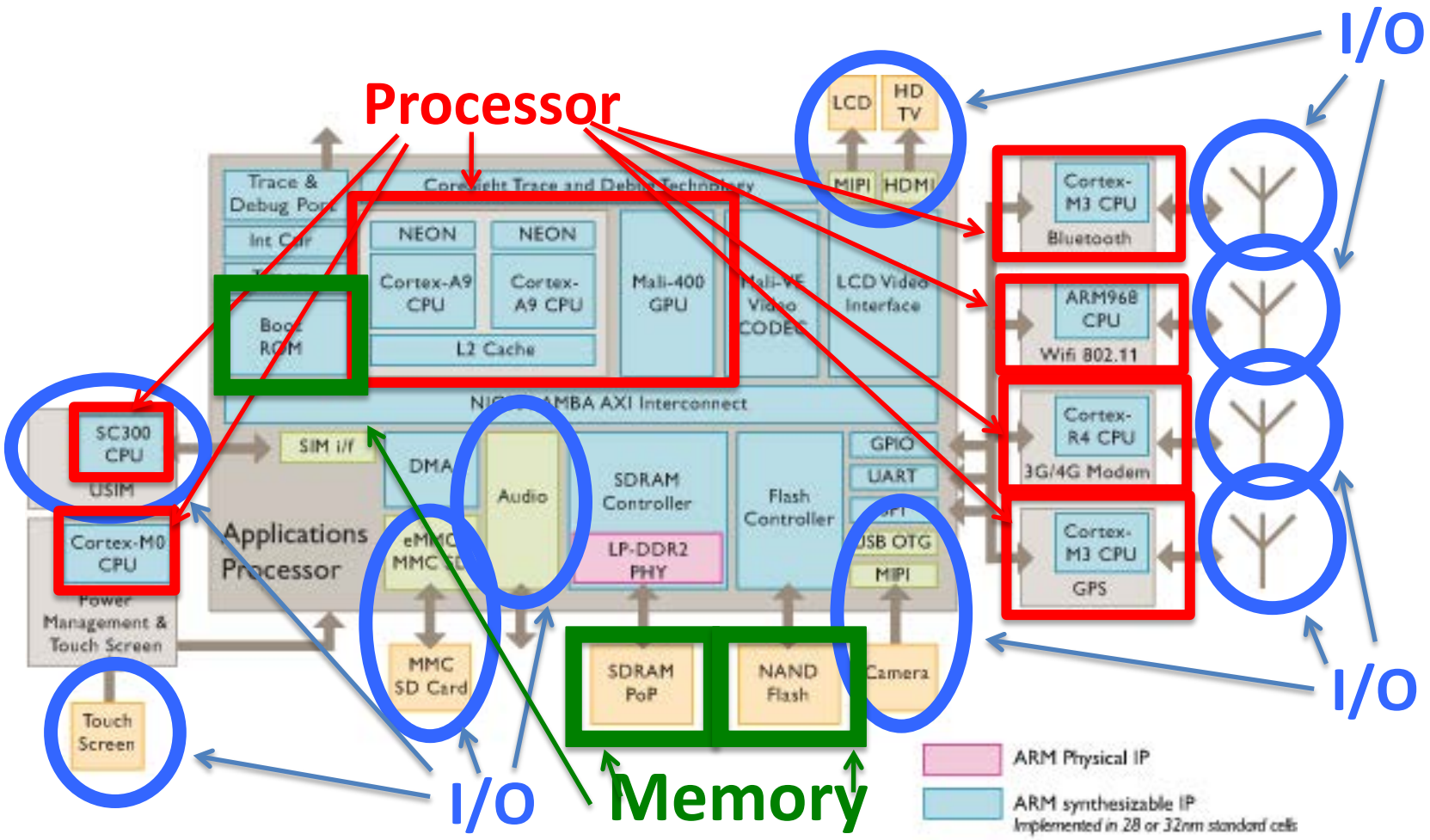
Advanced RISC Machine (ARM) instruction set inside the iPhone



You will learn how to design and program a related RISC computer: MIPS

instruction set：指令集；Cahce：高速缓存

iPhone Innards



You will learn about MIPS processor, cache, Memory, I/O in this course

What is this course about?

- It is about the hardware-software interface(接口)
  - What does the programmer need to know to achieve the highest possible performance(性能)?
- Use low-level programming languages (closer to underlying hardware)
  - Allows us to talk about key hardware features in higher-level terms
  - Allows programmers to harness underlying hardware parallelism(并行性) for high performance
- 设计一个功能型小计算机
  - 包括MIPS processor、memory、I/O。。。

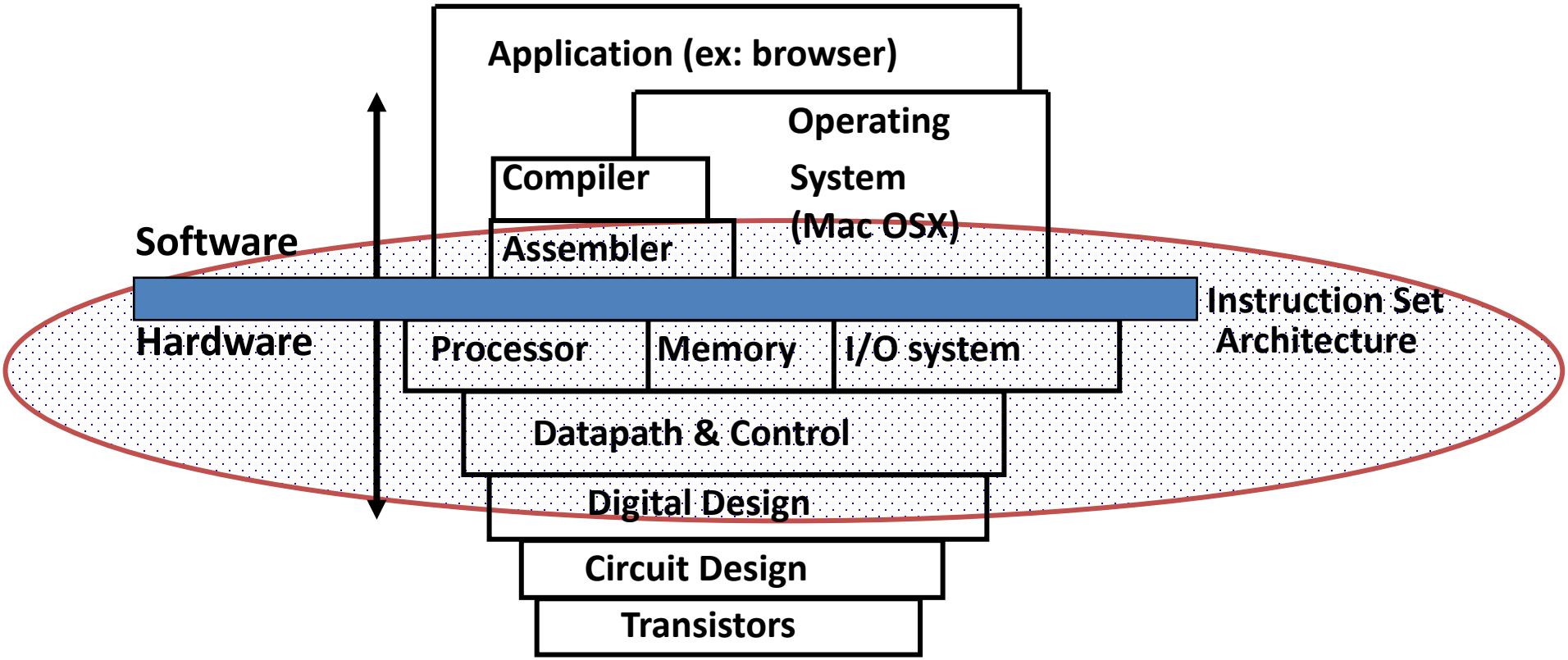
Why Not 80x86 vs. MIPS?

- Once learn one, easy to pick up others
- 80x86 instruction set is not beautiful
  - ≈ Full suitcase then add clothes on way to plane
  - Class time precious; why spend on minutiae?
- MIPS represents energy efficient processor of client (PostPC era) vs. fast processor of desktop (PC era)
- MIPS represents more popular instruction set:

  2010: 6.1B ARM, 0.3B 80x86 (20X more)

energy efficent：效能

Machine Structures



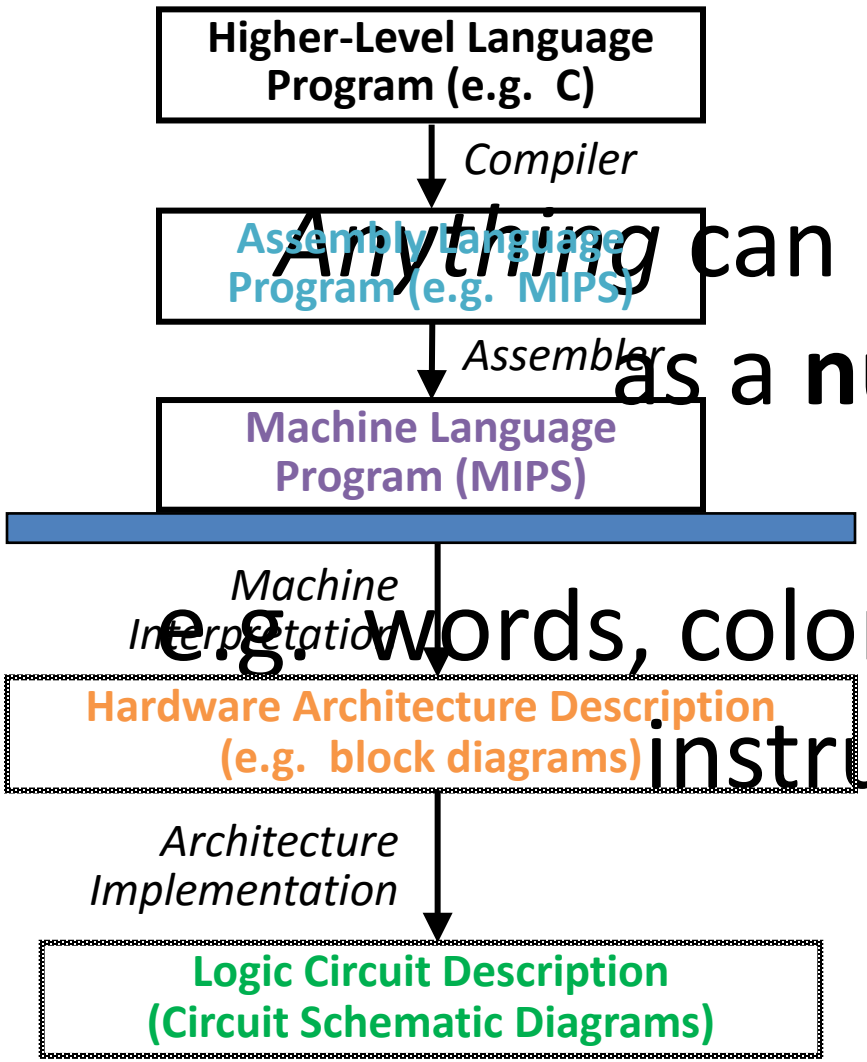
datapath/数据通路 assembler/汇编器 application/应用

Six Great Ideas in Computer Architecture

1. Layers of Representation/Interpretation
2. Moore's Law
3. Principle of Locality/Memory Hierarchy
4. Parallelism
5. Performance Measurement & Improvement
6. ~~Dependability via Redundancy~~

layer/层次 locality/局部性 memory hierarchy/存储层次 measurement/度量

Great Idea #1: Levels of Representation/Interpretation

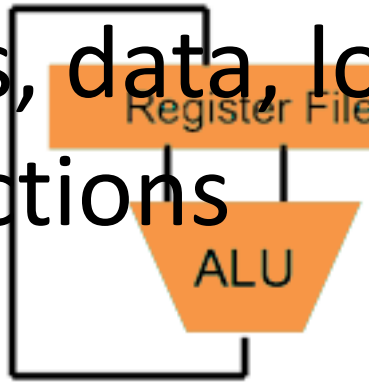


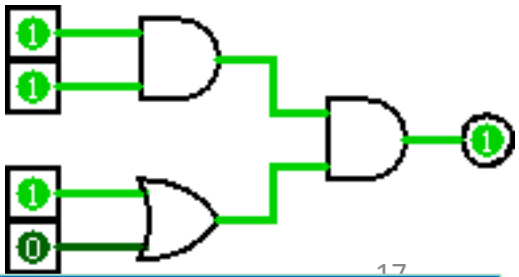

```
temp = v[k];
v[k] = v[k+1];
v[k+1] = temp;

lw   $t0, 0($2)
lw   $t1, 4($2)
sw   $t1, 0($2)
sw   $t0, 4($2)
```

```
0000 1001 1100 0110 1010 1111 0101 1000
1010 1111 0101 1000 0000 1001 1100 0110
1100 0110 1010 1111 0101 1000 0000 1001
0101 1000 0000 1001 1100 0110 1010 1111
```

Anything can be represented as a **number**!

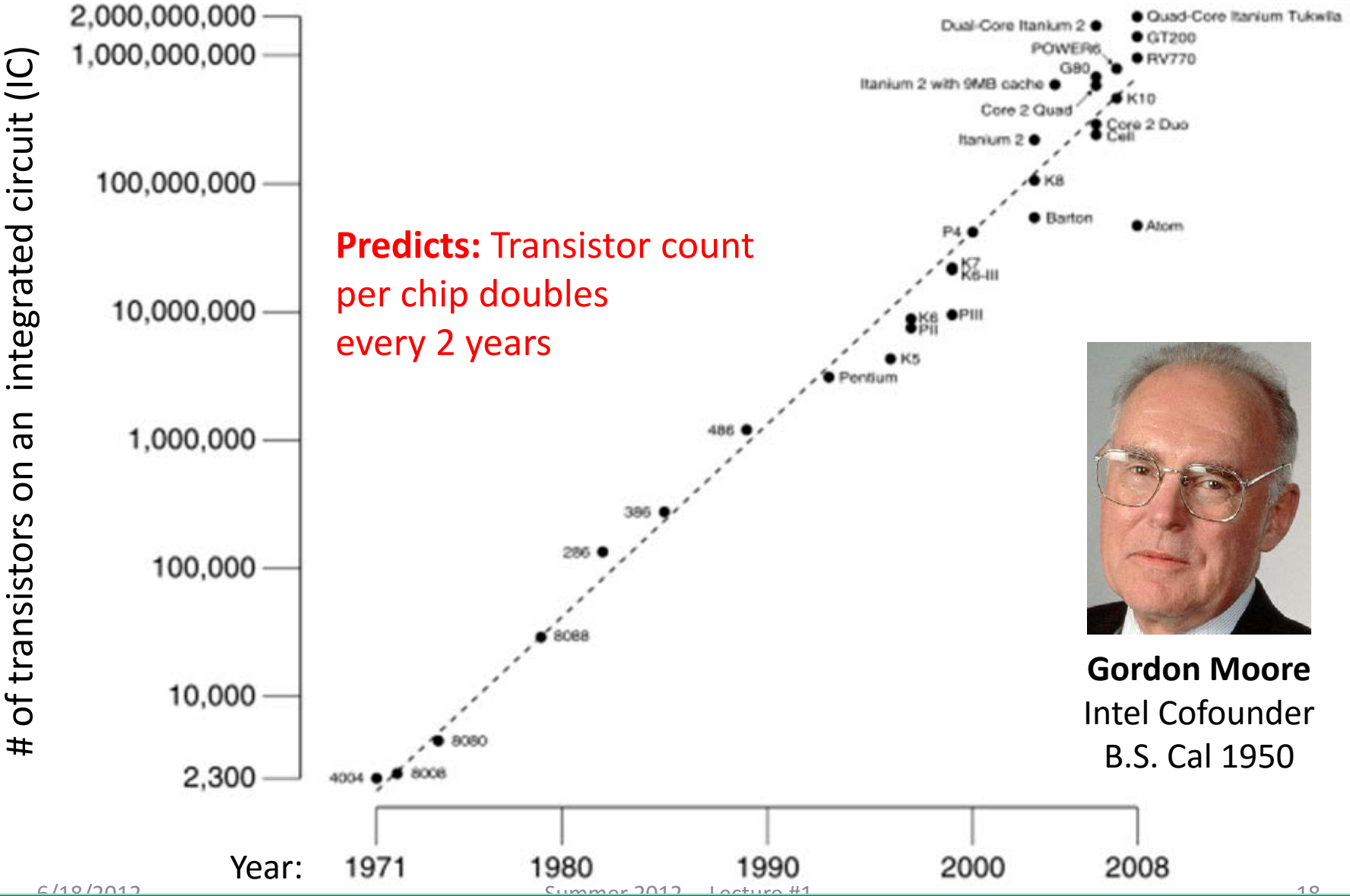
e.g. words, colors, data, logic, and instructions





representation/表示 interpretation/解释 schematic diagram/原理图

Great Idea #2: Moore's Law

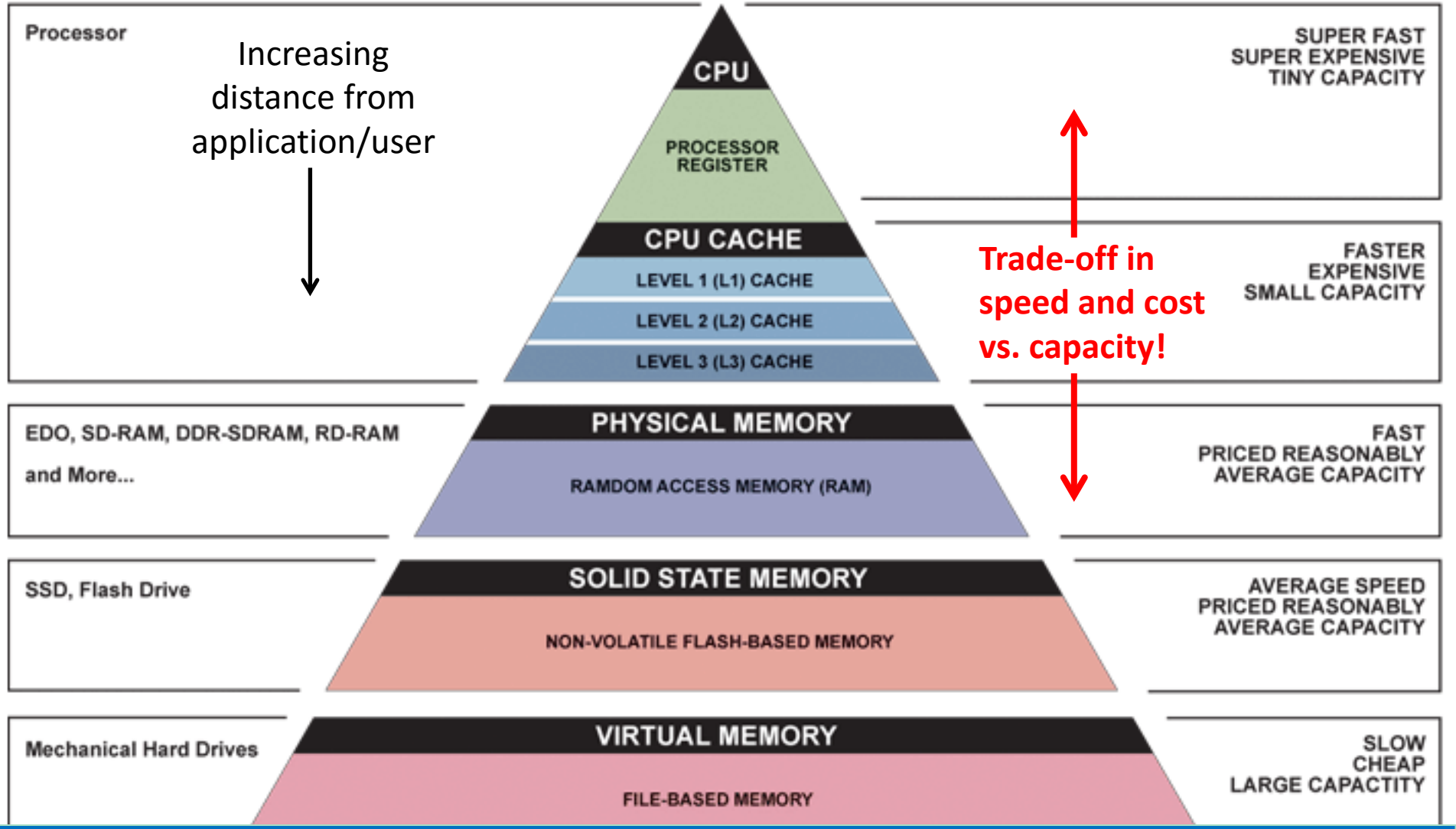


Predicts: Transistor count per chip doubles every 2 years

Gordon Moore
Intel Cofounder
B.S. Cal 1950

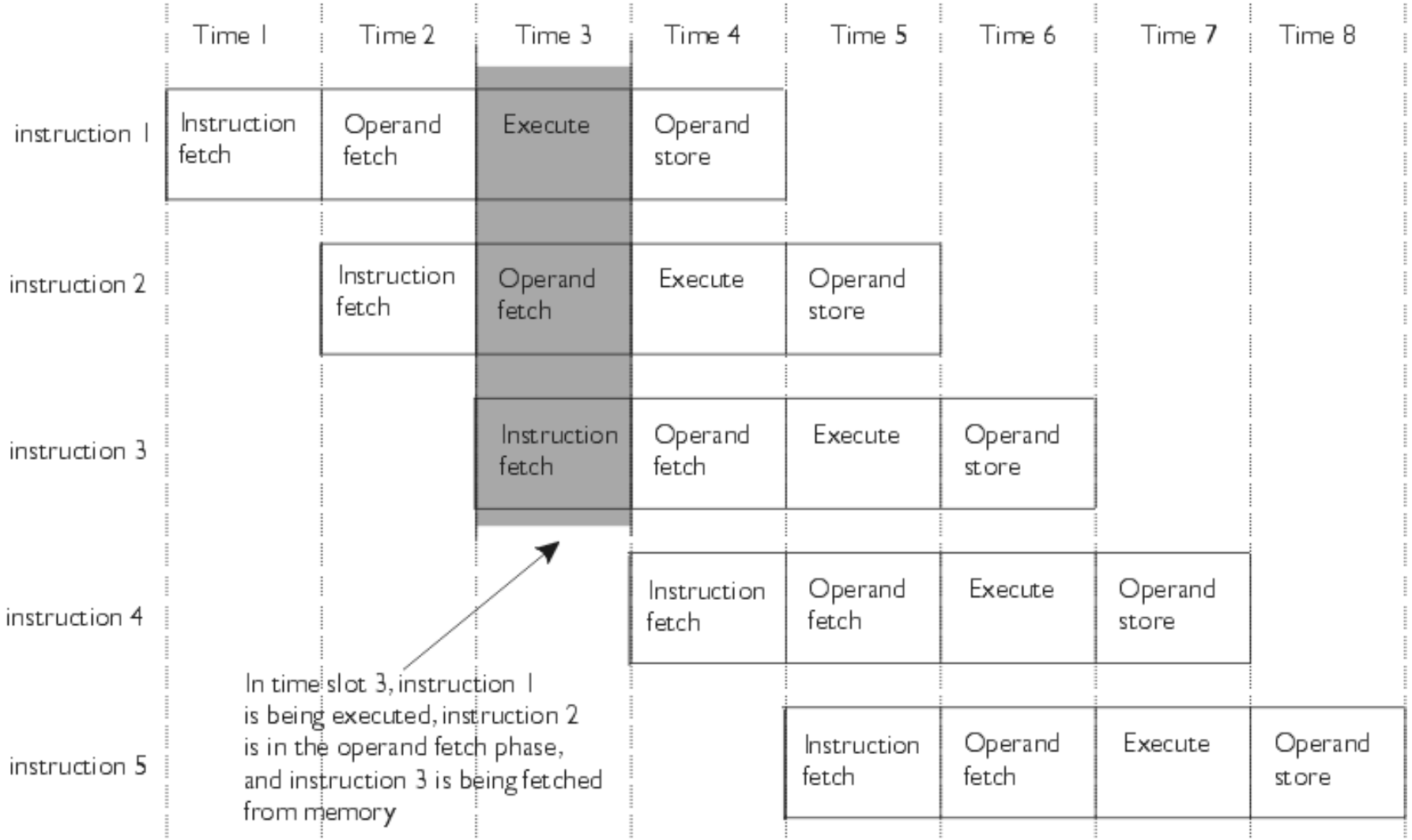
transistor/晶体管

Great Idea #3: Principle of Locality/ Memory Hierarchy



virtual memory/虚拟存储器 solid state memory/固态存储器 tradeoff/权衡(折衷)

Great Idea #4: Parallelism



parallelism/并行性

Great Idea #5: Performance Measurement and Improvement

- Allows direct comparisons of architectures and quantification of improvements
- It is all about *time to finish* (latency)
  - Includes both *setup* and *execution*.
- Match application and hardware to exploit:
  - Locality
  - Parallelism
  - Special hardware features, like specialized instructions (e.g. matrix manipulation)

latency/延迟 setup/建立 locality/局部性

Aside: Amdahl's Law

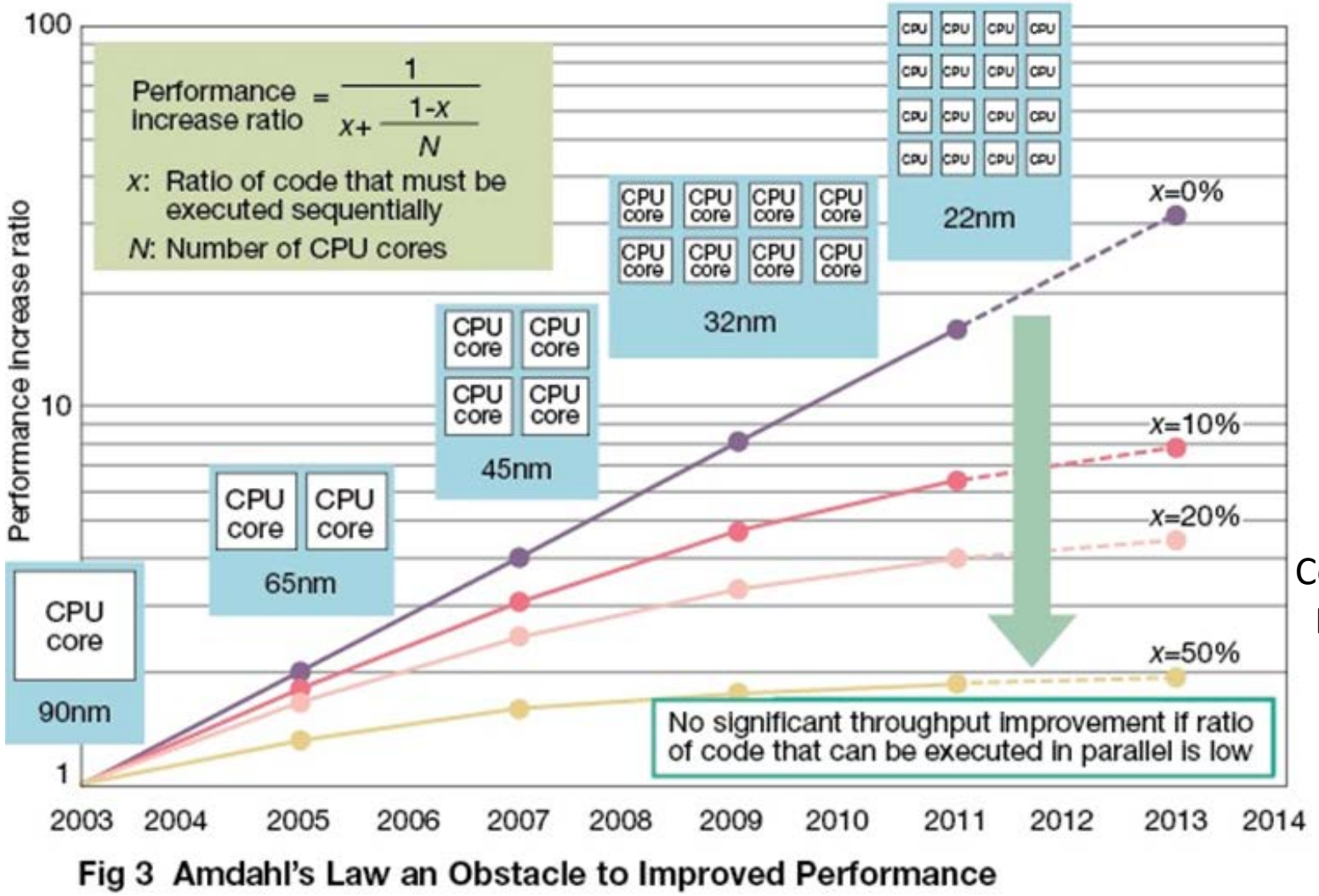


Fig 3 Amdahl's Law an Obstacle to Improved Performance

Gene Amdahl
Computer Pioneer
Ph.D. Wisconsin
1952

提纲

- 内容主要取材于CS61C第1讲
  - http://inst.eecs.berkeley.edu/~cs61c/su12
- 计算机基本认识
- 数制概念
  - Number Bases
  - Signed vs. Unsigned
  - Sign Extension

Number Representation

- Great Idea #1: Levels of Interpretation/Representation

- Inside a computer, everything stored as a sequence of 0's and 1's (bits)
  - Even this is an abstraction!
- How do we represent numbers in this format?
  - Let's start with integers

Number Bases

- **Key terminology:** digit (*d*) and base (*B*)
- Value of *i*-th digit is $d \times B^i$ where *i* starts at 0 and increases from right to left
  - *n* digit number $d_{n-1}d_{n-2} \ldots d_1d_0$
  - value = $d_{n-1} \times B^{n-1} + d_{n-2} \times B^{n-2} + \ldots + d_1 \times B^1 + d_0 \times B^0$
- In base B, each digit is one of B possible symbols
- Base is notated either as a prefix or subscript

Commonly Used Number Bases

- **Decimal** (base 10)
  - Symbols: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
  - Notation: 9472_{ten} = 9472
- **Binary** (base 2)
  - Symbols: 0, 1
  - Notation: 101011_{two} = 0b101011
- **Hexadecimal** (base 16)
  - Symbols: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F
  - Notation: $2A5D_{hex}$ = 0x2A5D

Decimal	Binary	Hex
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Number Base Examples

- Examples:

3 2 1 0

$\mathbf{9472}_{ten}$ = 9000 + 400 + 70 + 2

9x1000 + 4x100 + 7x10 + 2x1

$9x10^3 + 4x10^2 + 7x10^1 + 2x10^0$

$\mathbf{9472}_{ten} = 2x16^3 + 5x16^2 + 0x16^1 + 0x16^0$

$= \mathbf{2500}_{hex}$

0xA15 = 0b 1010 0001 0101

Bits Can Represent Anything

- *n* digits in base *B* can represent at most B^n things!
  - Each of the *n* digits is one of *B* possible symbols
  - Have more things? Add more digits!
- Example: Logical values (1 bit) – 0 is False, 1 is True
- Example: Characters
  - 26 letters require 5 bits ($2^5 = 32 > 26$)
- Example: Students in this class (? bits)

- For convenience, can group into *nibbles* (4 bits) and *bytes* (8 bits)

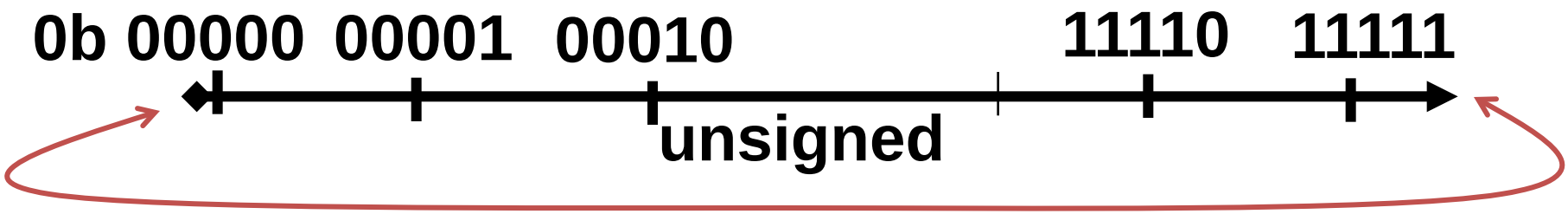
Unsigned Integers

Represent non-negative (unsigned) integers using base 2:

$0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000_{two} = 0_{ten}$
$0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0001_{two} = 1_{ten}$
$0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0010_{two} = 2_{ten}$
... ...
$0111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1101_{two} = 2{,}147{,}483{,}645_{ten}$
$0111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1110_{two} = 2{,}147{,}483{,}646_{ten}$
$0111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111_{two} = 2{,}147{,}483{,}647_{ten}$
$1000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000_{two} = 2{,}147{,}483{,}648_{ten}$
$1000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0001_{two} = 2{,}147{,}483{,}649_{ten}$
$1000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0010_{two} = 2{,}147{,}483{,}650_{ten}$
... ...
$1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1101_{two} = 4{,}294{,}967{,}293_{ten}$
$1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1110_{two} = 4{,}294{,}967{,}294_{ten}$
$1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111_{two} = 4{,}294{,}967{,}295_{ten}$

Overflow

- Numbers really have ∞ digits, but hardware can only store a finite number of them (fixed)
  - Usually ignore *leading zeros*
  - Leftmost is *most significant bit* (MSB)
  - Rightmost is *least significant bit* (LSB)
- Overflow is when the result of an arithmetic operation can't be represented by the hardware bits



most significant bit/最高有效位 overflow/溢出 arithmetic/算数

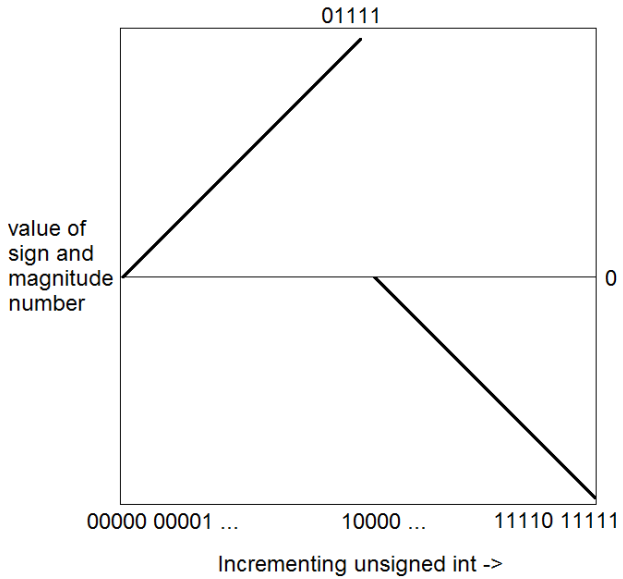
Signed Integers

- Programs often need to deal with negative numbers, so how do we encode these?
- *n* bits can represent 2^n different things
  - Ideally, want the range evenly split between positive and negative
- Can we encode them in such a way that we can use the same hardware regardless of whether the numbers are signed or unsigned?

negative/负 encode/编码

Sign and Magnitude

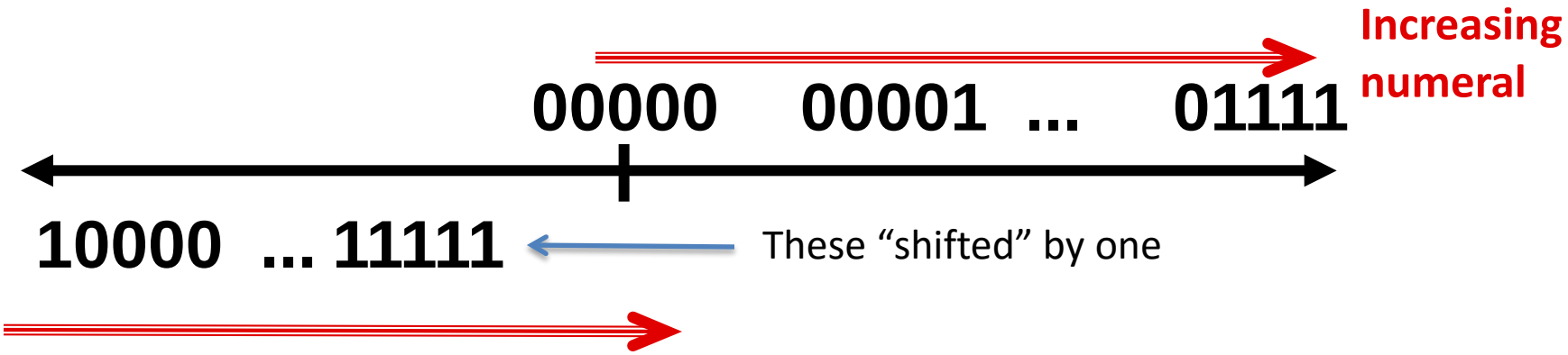
- MSB gives sign: 0 is positive, 1 is negative, rest of bits treated as unsigned (magnitude)
  - Examples: 0b 1000 0010 = -2, 0b 0000 0111 = 7
- Two zeros! 0b00…0 (+0) and 0b10…0 (-0)
- Cannot reuse unsigned hardware



magnitude~幅值/量级

Two's Complement

- Minor modification of one's complement
  - "Shift" representation of negative numbers down by one to remove duplicate zero



- Using this representation, incrementing the numeral *always* increments the integer
- To negate: complement the bits and add 1

complement~补码

Two's Complement

Sign Bit

$0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000_{two} = 0_{ten}$
$0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0001_{two} = 1_{ten}$
$0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0010_{two} = 2_{ten}$
... ...
$0111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1101_{two} = 2{,}147{,}483{,}645_{ten}$
$0111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1110_{two} = 2{,}147{,}483{,}646_{ten}$
$0111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111_{two} = 2{,}147{,}483{,}647_{ten}$
$1000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000_{two} = -2{,}147{,}483{,}648_{ten}$
$1000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0001_{two} = -2{,}147{,}483{,}647_{ten}$
$1000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0010_{two} = -2{,}147{,}483{,}646_{ten}$
... ...
$1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1101_{two} = -3_{ten}$
$1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1110_{two} = -2_{ten}$
$1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111_{two} = -1_{ten}$

2的补码表示法

- 假设$[A]_{补}$由$a_{n-1}a_{n-2}\cdots a_1a_0$表示
- 最高位a_{n-1}为符号位
  - a_{n-1}= 0 表示A为正数
  - a_{n-1}= 1 表示A为负数
- A取值范围：$[-2^{n-1}, 2^{n-1}-1]$

$$A = -2^{n-1}a_{n-1} + \sum_{i=0}^{n-2} 2^i a_i$$

$$1)\ A \geq 0 (即 a_{n-1} = 0)\ A = \sum_{i=0}^{n-2} 2^i a_i$$

$$2)\ A < 0 (即 a_{n-1} = 1)\ A = -2^{n-1} + \sum_{i=0}^{n-2} 2^i a_i$$

[X]补与[−X]补

若$[x]_{补}=x_0x_1x_2\ldots x_{n-1}$

则$[-x]_{补}=\overline{x}_0\overline{x}_1\overline{x}_2\ldots\overline{x}_{n-1}+1$

补码体系：-X = X各位取反 + 1

Two's Complement Summary

- Used by all modern hardware
- Roughly evenly split between positive and negative
  - One more negative # because positive side has 0
- Can still use MSB as sign bit
- To negate: Flip the bits and add one
  - Example: +7 = 0b 0000 0111, -7 = 0b 1111 1001

Two's Complement Review

- Suppose we had 4 bits. What integers can be represented in two's complement?

a) -15 to +15 ← need 5 bits

c) 0 to +15 ← unsigned

d) -8 to +7 ← two's complement

e) -16 to +15 ← need 5 bits

Sign Extension

- Want to represent the same number using more bits than before
  - Easy for positive #s (add leading 0's), more complicated for negative #s
  - Two's complement: copy MSB
- Example:
  - Two's complement: 0b 11 = 0b 1111

sign extension~符号扩展